INTL-0536-US
(P10841)

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: PROVIDING MULTIPLE SYMMETRICAL FILTERS

INVENTOR: DEAN ROSALES

Express Mail No. EL732849835US

Date: March 29, 2001

PROVIDING MULTIPLE SYMMETRICAL FILTERS

Background

This application relates generally to digital image processing.

Digital image processing may involve pixel group processing. Digital image processing may be used in cameras, scanners, and video applications, as examples. Pixel group processing involves a group of proximate pixels surrounding a center pixel. The pixels that surround the center pixel may provide information about brightness in a processed region. This brightness information may be utilized in connection with spatial filtering.

Any image may be composed of a plurality of frequency components including both low, high and intermediate frequencies. High spatial frequency components generally involve rapid brightness transitions. Low spatial frequency components involve slowly changing brightness transitions. Generally high frequencies may be found in association with sharp edges or points where white to black transitions occur within one or two pixels, as two examples.

An image may be filtered to accentuate or reduce any particular band of spatial frequencies such as high or low frequencies. Digital image processing techniques for removing particular frequency bands may be called spatial filtering operations.

Spatial filters may be implemented using a spatial convolution. A spatial convolution calculates what is going on with the pixel brightness around a particular processed pixel. The spatial convolution process may use a finite impulse response (FIR) filter.

A spatial convolution process may move across a representation of an image, pixel-by-pixel, placing resulting pixels in an output image. The brightness of each output pixel may depend on the input pixels that surround a particular processed pixel. A spatial convolution may calculate a spatial frequency in the area of a central pixel. As a result, the convolution may be capable of achieving filtering based on a region's spatial frequency content.

A spatial convolution may use a mathematical construction of the input pixel and its neighbors to determine an output pixel brightness value. A kernel is a group of pixels used to determine a mathematical calculation of output pixel brightness values. Kernel dimensions are generally that of a square with an odd number of values in each dimension. Thus, a kernel may have dimensions of 1x1, which would amount to a point, 3x3, 5x5 and so on. The greater the kernel size, the more neighboring pixels that are used in the calculation of the pixel brightness value.

A weighted average calculation is called a linear process. A weighted average may be utilized to determine the pixel brightness value. A linear process involves a

summation of elements multiplied by constant values. The elements are pixel brightnesses in the kernel and the constant values are the weights or convolution coefficients.

Taking as an example a 3x3 kernel convolution, a center pixel is evaluated in view of its eight neighbors. A convolution mask may be utilized to produce an output pixel value. The center pixel and its eight neighbors are multiplied by their respective convolution coefficients and the multiplicands may be summed. The result is placed in the output image at the same center pixel location. The process continues pixel-by-pixel for the pixels in a representation of an input image.

Thus, the equation for the spatial convolution process is follows:

$$O(x,y) = aI(x-1, y-1) + bI(x,y-1) + cI(x+1,y-1) + dI(x-1,y) + eI(x,y) + fI(x+1,y) + gI(x-1,y+1) + hI(x,y+1) + iI(x+1,y+1).$$

Common spatial filtering operations may include high-pass, low-pass and edge enhancement filters. Other filtering operations may be utilized as well. A problem arises in that different kernel sizes may be needed in different situations. Conventional convolution masks generally only determine a single kernel size. In some cases, a plurality of difference kernels may be needed. For example, in some cases, issues arise with respect to pixels closer to and/or further away from the current pixel.

Thus, there is a need for a system which provides multiple symmetrical filters at the same time.

Brief Description of the Drawings

Figure 1 is a block depiction of one embodiment of the present invention.

Figure 2 is a functional depiction of one embodiment of the present invention;

Figure 3 is a block diagram showing hardware for implementing the function illustrated in Figure 2 in accordance with one embodiment of the present invention;

Figure 4 is a functional depiction of one embodiment of the present invention;

Figure 5 is a block diagram depicting hardware for implementing the function shown in Figure 4 in accordance with one embodiment of the present invention;

Figure 6 is a block diagram showing hardware in accordance with one embodiment of the present invention; and

Figure 7 is a flow chart for software in accordance with one embodiment of the present invention.

Detailed Description

A plurality of symmetrical filters may be calculated simultaneously so that kernels of different sizes are available as needed. In one embodiment, an nxn kernel may be folded along the row and column directions to produce a compacted kernel. This compacted kernel may be subjected

to additions and multiplications in accordance with a spatial filtering algorithm in a way that enables outputs of a variety of desired kernel sizes.

Thus, a plurality of filters of different sizes may be determined from the same image data, for example in 3x3, 5x5, 7x7, 9x9 and 11x11 sizes. These sizes may be desirable for enabling spatial filtering with selectable degrees of freedom.

For example, in pixel group processing, the larger the size of the kernel of pixels used in the calculation of the weighted average of the input pixel brightness values, the greater the degree of freedom of the spatial filter. For example, the flexibility and precision of a spatial filter may be improved when more neighboring pixels are utilized in the calculation. Thus, the degree of precision or the degree of flexibility may be balanced with the speed of calculation, for example to select one of a variety of available kernel sizes for use in spatial filtering in a given case.

Referring to Figure 1, image data may be captured in a variety of different ways. For example, a digital imaging system may capture image data in a form of brightness values for an array of pixels that together may be utilized to recreate an image. The image data may be stored in a storage and adder hardware 10. Storage and adder hardware 10 may be responsible for adding together symmetrical rows of data to reduce the image data matrix size. Then the image data may be stored in a storage and adder hardware 16

which likewise reduces the matrix size by adding together symmetrical columns of image data.

Next, an adder accumulator section 28 and a multiply accumulate section 30 perform addition and multiplication sequences needed to implement a linear process or spatial convolution. Finally an adder tree 32 may be utilized to complete the calculation and to output a plurality of different kernel or matrix sizes.

In the illustrated embodiment, four kernel sizes are provided including 3x3, 5x5, 7x7, 9x9, and 11x11 matrices. Each of these output matrices may be selectably utilized, for example depending on the degree of freedom desired for a particular system. Because a plurality of different symmetrical filters are available at any given time, the flexibility of the system may be significantly increased in some cases. Moreover, the computational complexity is not proportionally increased because many components and calculations may be reused in each kernel calculation.

Each of the units 10, 16, 28, 30 and 32 may be operated under control of a state machine 11. The state machine 11 may provide information about where data should be stored, where data should be acquired, and what operations may be used to derive a desired result. The state machine 11 may be in turn controlled by a controller 15 such as a processor-based system. The state machine 11 may also include a storage 13. The storage 13 may include separate storage areas in one embodiment for storing the folded row and the folded column information.

The state machine 11 may be controlled by the controller 15 to provide a desired number of output kernel sizes. In addition, in some cases the state machine 11 may be controlled by the controller 15 to stop the determination of the kernel sizes. For example, in one embodiment, the kernels are produced one after the other from smaller to larger sizes. If a sufficient kernel is obtained, the state machine 11 may be stopped by the controller 15 to save time and to go on to other calculations. In other cases, it may be necessary to have a variety of kernel sizes. The state machine 11 may be programmed to output the desired array of kernel sizes.

Referring to Figure 2, an 11x11 array of image data may reside in the storage and adder hardware 10. The array may be read columnwise. The columns are indicated as C1 through C11. After an initial set of filters is calculated, a subsequent column is read and a center pixel moves to the right by one. The process may be repeated until reaching the final column of data. Thus, an 11x11 window slides across the original page of data.

As shown in Figure 3, the symmetry in the data may be exploited to pre-add pixel data that will be multiplied by the same convolution coefficient. For example, because of symmetry, the diagonals may all use the same convolution mask coefficients (1, 2, … 11) in one embodiment. The symmetry may be used to reduce the number of rows of data. Thus, as illustrated, row 11 (R11) is added to row 1 (R1),

row 10 (R10) is added to row 2 (R2), row 9 (R9) is added to row 3 (R3), etc. Row 6 (R6) is not added to any other row.

The row pre-adder 10 adds the rows together as the data is clocked into a hardware unit. The result is a 6x11 kernel. Row 6 (R6) remains as in the original data since it did not have a corresponding row to be added with.

Thus, row 1 (R1) and row 11 (R11) are added by adder 14a, row 2 (R2) and row 10 (10) are added by adder 14b, etc. The new array includes only six rows but still includes eleven columns (C1 through C11).

Referring next to Figure 4, the columns may be pre-added together to reduce the number of columns. Again the reduction is possible because of the inherent symmetry of the matrix. Thus, column 11 (C11) may be added to column 1 (C1), column C10 (C10) may be added to column 2 (C2) and so. Again, column 6 (C6) has no associated column and therefore it is not added to any other column.

Turning next to Figure 5, the column pre-adder circuitry 16 includes a plurality of adders 20 which receive the appropriate columns from the memory 10 and add them together. The six rows and six columns are then stored in the hardware 16. The storage area for holding the 6x6 matrix may be different from the 6x11 storage area. The value of using a separate storage area arises from the usefulness of reusing the data currently being held in the 6x11 storage area. Since the filter window is sliding across a page of data, new columns of data need to be read in, in order to calculate the next set of filters. As a

result, the 6x11 matrix is fed the new column of data and one column of old data is discarded.

Referring next to Figure 6, the filter values for a plurality of different filter kernel or matrix sizes are then calculated. Firstly, the values from the storage 26 in the hardware 16, that are needed to calculate a given kernel size, are identified. For a given symmetrical kernel size, some of the data values can be added together prior to multiplying with the corresponding convolution coefficient. In one example, because of symmetry, data values that lie in any given diagonal may be multiplied by the same coefficient. Thus, the diagonal 6 is from row 1, column 6 to row 6, column 1 and the diagonal 7 is from row 2, column 6 through row 6, column 2, etc.

In one embodiment, the filters may be calculated in order from the smaller to larger kernel size. Thus, in the illustrated embodiment, the 6x6 pre-added data storage 26 is utilized to first calculate the 3x3 matrix using row 5, columns 5 and 6 and row 6, columns 5 and 6 and then progressing as illustrated in Figure 6 through the 5x5, 7x7, 9x9 and 11x11 matrices.

Due to the symmetry, some intermediate values calculated for each of the filter sizes may be reused in calculating a subsequent filter. Reuse may reduce the number of calculations needed to perform the larger filter calculations, reducing the number of clocks to perform the overall filter calculation. The state machine 11 implements the re-use of intermediate values.

To calculate a given filter, some of the values in a given diagonal may be added together and accumulated in the adder accumulator (AAC) section 28 in one embodiment. These results may be saved for use in subsequent filter calculations.

The calculation of the 3x3 filter utilizes the data in the box labeled 3x3 in the storage 26. The pre-added data value sitting in row 6, column 6 (the data value in diagonal 11) is directed by the state machine 11 to the register 36k in the section 28. No adding by an adder 34 is needed because the diagonal 11 includes only a single pre-added value.

The pre-added data in diagonal 10 may use one of the AAC section 28 adders 34 and the result is stored in the holding register 36j. In particular, the state machine 11 causes a specified adder, such as the adder 34e, to add the values in diagonal 10 (row 6, column 5 and row 5, column 6) and to place the result in the register 36j, for example. The data value in row 5, column 6 that belongs to diagonal 9 is moved directly to the accumulator storage area 36i, again because no adding is necessary.

Three multipliers, such as multipliers 38c, 38d and 38e, are used to perform the multiplication of the three data values in the registers 36i-k with their respective convolution coefficients from the coefficient bank 48. The multiplication results from the multipliers 38d and 38e are added by the adder 42b and that result is added, in the adder 46, to the result of the multiplier 38c. Each of the

components shown in Figure 6 may be controlled by the state machine 11.

To calculate the 5x5 filter, the data elements contained within the box labeled 5x5 in the storage 26 are used. Starting at the lower right hand corner of the data storage 26 and moving up, the data value on the diagonals 11 and 10 are already sitting in their respective accumulators ready for multiplication due to the prior calculation of the 3x3 filter. The values for diagonal 9 need to be added together. One of the data points, row 5, column 5, is sitting in the accumulator section 28. The remaining two values, in row 4, column 6 and row 6, column 4, need to accumulated. One of the accumulator section adders 34 may be utilized for this function and the result may be stored in a register 36i.

There are two values for diagonal 8 (row 4, column 5 and row 5, column 4) that need to be added together and stored in the register 36h. One of the adder accumulator section adders 34 may be used to accomplish this task with the result being saved in register 36h.

The value for diagonal 7 (row 4, column 4) may be moved into register 36g. The multiply accumulate section 30 may be utilized to perform the final multiplication step. The values sitting in the registers 36g through 36k are multiplied by their corresponding coefficients from the coefficient bank 48. The coefficients used for the 5x5 matrix are not necessarily the same as those used for the 3x3 matrix.

The remaining filters may be calculated the same way. The larger filters may require more multiplications to complete the filtering process. The multiply accumulate section 30 may be used to calculate portions of the filter and subsequently to calculate the remaining data points.

The matrix shown in Figure 2 has a symmetry not only about row 6 and column 6 but also a symmetry along the diagonals. Embodiments of the present invention may be utilized with a variety of other types of symmetry. Other potential symmetry includes symmetry about a middle row and column but with no diagonal symmetry. In some cases, the difference in symmetry may necessitate differences in the calculations.

In some embodiments of the present invention, many of the steps are done in parallel, reducing the overall number of clocks needed to generate a result. The calculations may be pipelined in some embodiments and many steps may be accomplished in one clock cycle. Latencies may begin to accumulate when the multiply accumulate operations occur. The multiply accumulate latencies occur when calculating larger filters.

Thus, in one embodiment, adding the rows together may take five clocks, adding the columns together may take five clocks, the calculation of a 3x3 matrix may take one clock, a 5x5 matrix may take one clock, a 7x7 matrix may take two clocks, a 9x9 matrix may take two clocks and the 11x11 matrix may take three clocks. Thus the initial estimate of overall filter performance may be viewed as calculating all

filters every ten clocks. Estimates for individual filters may vary and may be determined on a case by case basis.

Performance may be based on a given filter's symmetry, the amount of available hardware resources, and the kernel sizes specified. Storage areas 15 and 17, shown in Figures 2 and 4, may be made of discrete registers, utilizing random access memory with appropriate controls in one embodiment. This implementation may reduce the number of gates but may decrease the overall performance of the filter. The number of multiply accumulate devices 40 may be reduced to save gates but similarly the overall performance may be reduced.

While embodiments of the present invention have been described with respect to an 11x11 window, other window sizes may be utilized as well. Any of the variety of types of symmetry may be exploited. Any combination of one or more filters may be calculated simultaneously. The multiply accumulate section 30 may use floating point units in one embodiment to address additional applications.

Referring to Figure 7, the filter calculation software 50 may be executed by the controller 15 to control the state machine 11 in accordance with one embodiment of the present invention. In another embodiment of the present invention, the calculation of filter coefficients may be done in whole or in part using software routines. In such case, all or some of the hardware depicted in Figure 1 may be unnecessary and instead the controller 15 loaded with

the software 50 may be utilized to calculate the filter kernels.

Continuing in Figure 7, the image data is received as indicated at 52. Thereafter, the columns are added (block 54) and the rows are added (block 56) to reduce the matrix size in one embodiment. The convolution equation is executed for a first, smaller filter size as indicated in block 58. Then, the convolution equation is executed for a larger, second filter size (block 60). Calculations that were done for the first filter size may be maintained and reused during the calculation of the second filter.

At diamond 62 a check determines whether there are more filters to calculate. If so, the flow iterates. If not, the flow moves to the next adjacent pixel and the window moves likewise by one. The routine depicted in Figure 7 may then be repeated for the neighboring pixel. The software 50 may in one embodiment progressively output the smaller filter and then the larger filter. In addition, any number of filters may be progressively calculated.

While the present invention has been described with respect to a limited number of embodiments, those skilled in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this present invention.

What is claimed is: